FJ 02-028 US

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００２年　６月２７日

出願番号
Application Number: 特願２００２－１８７２３０

[ST.10/C]: [ＪＰ２００２－１８７２３０]

出願人
Applicant(s): 富士通株式会社

２００３年　１月　７日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証番号　出証特２００２－３１０３５６０

【書類名】　特許願

【整理番号】　0240751

【提出日】　平成14年 6月27日

【あて先】　特許庁長官殿

【国際特許分類】　G06F 15/16

【発明の名称】　ロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置

【請求項の数】　9

【発明者】

【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

【氏名】　三宅　英雄

【発明者】

【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

【氏名】　上方　輝彦

【発明者】

【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

【氏名】　畔上　謙吾

【特許出願人】

【識別番号】　000005223

【氏名又は名称】　富士通株式会社

【代理人】

【識別番号】　100104190

【弁理士】

【氏名又は名称】　酒井　昭徳

【手数料の表示】

【予納台帳番号】　041759

【納付金額】　　21,000円

【提出物件の目録】

【物件名】　　明細書　1

【物件名】　　図面　1

【物件名】　　要約書　1

【包括委任状番号】　9906241

【プルーフの要否】　　要

【書類名】　　明細書

【発明の名称】　ロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置

【特許請求の範囲】

【請求項１】　複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、

前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定する共有データ判定工程と、

前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに、所定の識別情報を付加する識別情報付加工程と、

前記識別情報付加工程で所定の識別情報を付加されたデータを結合してメモリ空間内のキャッシュ非対象領域に配置するための領域を形成する共有データ領域形成工程と、

を含んだことを特徴とするロードモジュール生成方法。

【請求項２】　前記識別情報付加工程では、前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに、識別情報として所定の接頭子を付加することを特徴とする請求項１に記載のロードモジュール生成方法。

【請求項３】　前記識別情報付加工程では、前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに、識別情報としてその所属するセクションを指定する情報を付加することを特徴とする請求項１に記載のロードモジュール生成方法。

【請求項４】　複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、

前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち

少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定する共有データ判定工程と、

前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに対してキャッシュ無効化操作を付加するキャッシュ無効化操作付加工程と、

を含んだことを特徴とするロードモジュール生成方法。

【請求項５】　前記キャッシュ無効化操作付加工程では、前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに対する読み出し命令の直前に、当該データのキャッシュのインヴァリデート命令を挿入することを特徴とする請求項４に記載のロードモジュール生成方法。

【請求項６】　複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成プログラムにおいて、

前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定する共有データ判定工程と、

前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに、所定の識別情報を付加する識別情報付加工程と、

前記識別情報付加工程で所定の識別情報を付加されたデータを結合してメモリ空間内のキャッシュ非対象領域に配置するための領域を形成する共有データ領域形成工程と、

をコンピュータに実行させることを特徴とするロードモジュール生成プログラム。

【請求項７】　複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成プログラムにおいて、

前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち

少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定する共有データ判定工程と、

前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに対してキャッシュ無効化操作を付加するキャッシュ無効化操作付加工程と、

をコンピュータに実行させることを特徴とするロードモジュール生成プログラム。

【請求項８】　複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成装置において、

前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定する共有データ判定手段と、

前記共有データ判定手段により少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに、所定の識別情報を付加する識別情報付加手段と、

前記識別情報付加手段により所定の識別情報を付加されたデータを結合してメモリ空間内のキャッシュ非対象領域に配置するための領域を形成する共有データ領域形成手段と、

を備えたことを特徴とするロードモジュール生成装置。

【請求項９】　複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成装置において、

前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定する共有データ判定手段と、

前記共有データ判定手段により少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに対してキャッシュ無効化操作を付加するキャッシュ無効化操作付加手段と、

を備えたことを特徴とするロードモジュール生成装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

この発明は、複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置に関する。

【０００２】

【従来の技術】

計算機システムにおいては、プログラム実行時の参照の局所性（ｌｏｃａｌｉｔｙ　ｏｆ　ｒｅｆｅｒｅｎｃｅ）を利用することでその処理能力の向上をはかるべく、プロセッサと主記憶との間にキャッシュ機構を設けることがある。

【０００３】

このキャッシュ機構は、主記憶よりは小さい容量だが高速に読み書き可能なキャッシュ・メモリを有し、プロセッサからの読み書き要求に応答して、キャッシュ・メモリあるいは主記憶の読み書きをおこなう。このとき、参照の局所性を活用すべく、一度読み書きした主記憶のメモリ領域の内容（値）をキャッシュ・メモリに複写しておく。複写されたメモリ領域に関しては、主記憶でなくキャッシュ・メモリを読み書きすればよいこととなり、処理の高速化が実現される。

【０００４】

ところで近年の計算機システムでは、複数のプロセッサを搭載し、プログラム中の各部分プログラムをこれらのプロセッサに配分することで処理能力の向上をはかる例がある。このような手法は「共有メモリ型マルチプロセッサ方式（Ｓｈａｒｅｄ－Ｍｅｍｏｒｙ　Ｍｕｌｔｉｐｒｏｃｅｓｓｏｒｓ）」と呼ばれている。そして、この共有メモリ型マルチプロセッサ方式においても、図１０に示すようにプロセッサごとにキャッシュ機構を設けることは、処理能力の向上をはかる上で有効である。

【０００５】

ただ、上記方式においては複数のキャッシュ・メモリを有することになるため、それぞれのキャッシュ・メモリと主記憶の間で、同一アドレスで特定されるメモリ領域の値が一致しなくなることがある。これは主記憶上の任意のメモリ領域に対する任意のプロセッサからの読み出しが、その領域に保持された最新の値を常に返すわけではないことを意味しており、「キャッシュの一致性問題（ｃａｃｈｅ　ｃｏｈｅｒｅｎｃｅ　ｐｒｏｂｌｅｍ）」として知られている。

【０００６】

そして、従来技術においてはこの問題は、「キャッシュ一貫性機構」と呼ばれる物理的な機構によりハードウエア的に解決されるのが通例であった。これは複数の部分プログラムにより読み書きされるデータ（以下では「共有データ」と総称する）の位置を監視し、更新前の古いデータがキャッシュされるのを防ぐキャッシュ一貫性プロトコル（ｃａｃｈｅ　ｃｏｎｓｉｓｔｅｎｃｙ　ｐｒｏｔｏｃｏｌ）にもとづいて、キャッシュの一貫性を保障するものである。

【０００７】

なお、図１１はキャッシュ一貫性機構によりキャッシュの一貫性を保障する場合のメモリマップ例を示す説明図である。図中、ｔｅｘｔ　ａｒｅａ１１００はプログラムの命令列を保持する領域、ｄａｔａ　ａｒｅａ１１０１はプログラムから読み書きされるデータ（共有データであると否とを問わない）を保持する領域である。

【０００８】

そして、これら二つの領域はいずれもキャッシュ対象領域、すなわちキャッシュ・メモリに複写される可能性のある領域である。したがって、共有データは当該データを共有する部分プログラムを実行中の、複数のプロセッサのキャッシュ・メモリにそれぞれ複写される可能性があるが、上述のキャッシュ一貫性機構により、各々のキャッシュ・メモリの値（および主記憶の値）が常に一致するように制御されるわけである。

【０００９】

【発明が解決しようとする課題】

しかしながらこうしたハードウエア的な手法では、キャッシュ一貫性機構その

ものが複雑であることから、プロセッサの回路規模が大きくなってしまうという問題点があった。

【００１０】

従来は、共有メモリ型マルチプロセッサ方式が採用されるのは主としてハイエンドの製品であったため、この点はあまり問題にならなかったが、今後一般向けのプリンタやデジタルカメラ、デジタルテレビなどに複数のプロセッサを搭載することを考えると、キャッシュの一貫性を保障するためだけにプロセッサが大きく／重くなったり、製品価格が上昇したりすることは避けなければならない。

【００１１】

この発明は上記従来技術による問題をソフトウエア的に解決するため、同一データを共有する複数の部分プログラムがそれぞれ別個のプロセッサにより実行されても、キャッシュの一貫性が自動的に維持されるようなロードモジュールを生成することが可能なロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置を提供することを目的とする。

【００１２】

【課題を解決するための手段】

上述した課題を解決し、目的を達成するため、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムまたはロードモジュール生成装置は、複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定し、そのようなデータであると判定されたデータには所定の識別情報を付加した上、この識別情報を付加されたデータを結合してメモリ空間内のキャッシュ非対象領域に配置するための領域を形成することを特徴とする。

【００１３】

また、この発明にかかるロードモジュール生成方法は、上記識別情報として共有データに所定の接頭子を付加することを特徴とする。

【００１４】

また、この発明にかかるロードモジュール生成方法は、上記識別情報として共有データの所属するセクションを指定する情報を付加することを特徴とする。

【００１５】

また、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムまたはロードモジュール生成装置は、複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定した上、そのようなデータであると判定されたデータに対してキャッシュ無効化操作を付加することを特徴とする。

【００１６】

また、この発明にかかるロードモジュール生成方法は、上記キャッシュ無効化操作として、共有データに対する読み出し命令の直前に、当該データのキャッシュのインヴァリデート命令を挿入することを特徴とする。

【００１７】

これらの発明によって生成されたロードモジュールの実行時には、複数の部分プログラムにより共有されるデータはそもそもプロセッサのキャッシュ・メモリに複写されないか、あるいは複写はされるが読み出し時には消去されているために、常に主記憶の値が参照・更新されることになる。

【００１８】

【発明の実施の形態】

以下に添付図面を参照して、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置の好適な実施の形態を詳細に説明する。

【００１９】

上述のように、従来技術ではキャッシュ一貫性機構というハードウエアを用いてキャッシュ・メモリー主記憶間、あるいはキャッシュ・メモリ相互間のデータ

の同一性を保障していたのであるが、本発明においてはキャッシュの一致性問題を、もっぱらソフトウエア的に解決する方針である。キャッシュの一貫性を、プログラムを実行するプロセッサ側でなく、プロセッサで実行されるプログラム側から保障しようとするアプローチと言ってもよい。

【００２０】

そして、この方針に沿う解決として理論上は下記二つの手法が提案されている（ｃｆ．シメル、カート「ＵＮＩＸ（Ｒ）カーネル内部解析－キャッシュとマルチプロセッサの管理」ソフトバンク社）。

【００２１】

（１）プログラム内の共有データはそもそもキャッシュしないようにする、すなわち常に主記憶上の値を読み書きすることで、共有データのコピーが複数箇所に散在する状況を未然に防止する方式。以下では「アンキャッシュ共有データ手法」と呼ぶ。

【００２２】

ＭＭＵ（Ｍｅｍｏｒｙ　Ｍａｎａｇｅｍｅｎｔ　Ｕｎｉｔ）を備えたプロセッサの中には、メモリ空間内の任意の領域をキャッシュ・メモリに複写しない領域として指定できるものがある。たとえばＳＰＡＲＣでは、ＭＭＵのページ・テーブル・エントリのＣビット（キャッシュ可能ビット）をＯＦＦにすることで、任意の領域をキャッシュ対象外とすることができる。

【００２３】

この機能は主に、メモリ空間とは別にＩ／Ｏ空間を設けるのでなく、メモリ空間の一部をＩ／Ｏ空間として使用する場合に、当該領域をキャッシュ対象外とする必要があることから用意されているものである。Ｉ／Ｏについては、キャッシュがあると最新の値が読み込めない／書き込めないなどの不具合が生ずるため、処理速度を犠牲にしてもキャッシュをしない設定としておくことに合理性がある。

【００２４】

この機能をいわば転用して、プロセッサのメモリ空間上にキャッシュ非対象領域を設け、共有データはもっぱらこの領域に配置するようにすれば、各プロセッ

サのキャッシュ・メモリには非共有データ、すなわち当該プロセッサで実行中の部分プログラムに固有のデータしか蓄積されないことになり、共有データについてはキャッシュが存在しないので、常に主記憶上の値が読み書きされることになる。

【００２５】

図１は、アンキャッシュ共有データ手法によりキャッシュの一貫性を保障する場合のメモリマップ例を示す説明図である。図中、ｔｅｘｔ　ａｒｅａ１００はプログラムの命令列を保持する領域、ｄａｔａ　ａｒｅａ１０１はプログラムから読み書きされるデータのうち、とくに非共有データを保持する領域である。そしてこれら二つの領域は、いずれもキャッシュ対象領域、すなわちキャッシュ・メモリに複写される可能性のある領域である。

【００２６】

これに対し、ｓｈａｒｅｄ　ｄａｔａ　ａｒｅａ１０２はプログラムから読み書きされるデータのうち、とくに共有データを保持する領域である。そして、この領域はキャッシュ非対象領域、すなわちキャッシュ・メモリに複写される可能性のない領域である。

【００２７】

以下で説明する実施の形態１および２は、この「アンキャッシュ共有データ手法」によりキャッシュの一貫性を保障しようとするものである。より具体的には上記を前提として、プログラムのロードモジュールの生成時に、メモリ空間内の異なる場所に配置される上記各領域をどうやって形成するかのその手順に関するものである。

【００２８】

（２）共有データも非共有データも区別なくキャッシュはするが、共有データについては読み出しの直前にキャッシュ・メモリ上のデータを無効化してしまうことで、常に主記憶上の値を読みに行き事実上キャッシュを無視する方式。以下では「選択的キャッシュ無効化操作手法」と呼ぶ。選択的と言うのは、非共有データについてはキャッシュの無効化をおこなわない点による。

【００２９】

キャッシュ機構には、ストア命令を実行しただけでキャッシュ・メモリの値とあわせて主記憶の値も同時に更新されるライトスルー・キャッシュと、キャッシュ・メモリにおける更新を主記憶にも反映させるには、ストア命令の後に改めてフラッシュ（ｆｌｕｓｈ）命令を実行しなければならないライトバック・キャッシュとの二種類がある。そして、キャッシュ機構がいずれのタイプであるかによって、上記の「無効化操作」の手順は異なる。

【００３０】

すなわちライトスルー型の場合は、主記憶には常に最新の値が保持されているので、共有データの読み出しに先立ってインヴァリデート（ｉｎｖａｌｉｄａｔｅ）命令を実行することで、自己のキャッシュ・メモリに残っているそのコピーを消去するだけでよい。これに対しライトバック型の場合は、まずフラッシュ命令を実行してキャッシュ・メモリ上の値で主記憶上の値を更新する、すなわち主記憶上の値を最新の値にアップデートした上で、インヴァリデート命令によりキャッシュを消去し主記憶上の値を読み込むという二段構えになる。

【００３１】

図２は、選択的キャッシュ無効化操作手法によりキャッシュの一貫性を保障する場合のメモリマップ例を示す説明図である。図中、ｔｅｘｔ　ａｒｅａ２００はプログラムの命令列を保持する領域、ｄａｔａ　ａｒｅａ２０１はプログラムから読み書きされるデータのうち、とくに非共有データを保持する領域、ｓｈａｒｅｄ　ｄａｔａ　ａｒｅａ２０２は共有データを保持する領域である。そして、これら三つの領域はいずれもキャッシュ対象領域である。

【００３２】

以下で説明する実施の形態３は、この「選択的キャッシュ無効化操作手法」によりキャッシュの一貫性を保障しようとするものである。より具体的には、プログラムのロードモジュールの生成時にあらかじめ共有データを特定しておき、当該データのロード命令の直前にキャッシュの無効化操作、すなわちライトスルー・キャッシュにおいてはインヴァリデート命令、ライトバック・キャッシュにおいてはフラッシュ命令＋インヴァリデート命令をそれぞれ挿入する、その手順に関するものである。

【００３３】

（実施の形態１）

図３は、本発明の実施の形態１にかかるロードモジュール生成装置のハードウエア構成の一例を示すブロック図である。

【００３４】

図中、まずＣＰＵ３０１は装置全体の制御を司る。ＲＯＭ３０２はブートプログラムなどを記憶している。ＲＡＭ３０３はＣＰＵ３０１のワークエリアとして使用される。ＨＤＤ３０４は、ＣＰＵ３０１の制御にしたがってＨＤ３０５に対するデータのリード／ライトを制御する。ＨＤ３０５は、ＨＤＤ３０４の制御にしたがって書き込まれたデータを記憶する。

【００３５】

ＦＤＤ３０６は、ＣＰＵ３０１の制御にしたがってＦＤ３０７に対するデータのリード／ライトを制御する。ＦＤ３０７は、ＦＤＤ３０６の制御にしたがって書き込まれたデータを記憶したり、記憶しているデータをＦＤＤ３０６の磁気ヘッドに読み取らせたりする。着脱可能な記録媒体としては、ＦＤ３０７のほかＣＤ－ＲＯＭ、ＣＤ－Ｒ、ＣＤ－ＲＷ、ＭＯ、ＤＶＤ（Ｄｉｇｉｔａｌ　Ｖｅｒｓａｔｉｌｅ　Ｄｉｓｋ）、メモリカードなどが考えられる。

【００３６】

ディスプレイ３０８は、たとえばＣＲＴ、ＴＦＴ液晶ディスプレイ、プラズマディスプレイなどであって、カーソルやウィンドウをはじめ、文書、画像などの各種データを表示する。ネットワークＩ／Ｆ３０９は、イーサネット（Ｒ）ケーブル３１０を通じてＬＡＮに接続されるとともに、ＬＡＮと装置内部とのデータの送受信を司る。

【００３７】

キーボード３１１は、文字、数値、各種指示などの入力のためのキーを備え、装置内部へのデータの入力をおこなう。タッチパネル式の入力パッドやテンキーなどであってもよい。マウス３１２は、カーソルの移動や範囲選択などをおこなう。ポインティングデバイスとして同様の機能を備えるものであれば、トラックボール、ジョイスティック、十字キー、ジョグダイヤルなどであってもよい。な

お、上記各部はバスまたはケーブル３００により接続されている。

【００３８】

次に、図４は本発明の実施の形態１にかかるロードモジュール生成装置の構成を機能的に示すブロック図である。同図に示す各機能部は、具体的には図３に示したＨＤ３０５、ＦＤ３０７などに格納されたプログラム、具体的にはコンパイラ、アセンブラおよびリンカの三つのプログラムを、ＣＰＵ３０１がＲＡＭ３０３に読み出して実行することにより実現される。

【００３９】

図中、４００～４０４は上記プログラムのうちコンパイラ、４０５～４０７はアセンブラ、４０８～４１２はリンカによりそれぞれ実現される機能部である。なお、各機能部の機能についてはすぐ下に述べるフローチャートで説明する。

【００４０】

次に、図５は本発明の実施の形態１にかかるロードモジュール生成装置におけるロードモジュール生成処理の手順を示すフローチャートである。

【００４１】

上記装置ではまずコンパイラが起動され、当該コンパイラにより実現される第１解析部４００が、指定されたプログラムのソース記述を読み込んで字句解析および構文解析をおこなうとともに、当該プログラムをコンパイラの内部表現へと変換する（ステップＳ５０１）。

【００４２】

次に共有データ判定部４０１が、コンパイラの内部表現を走査することで、そこに含まれる個々のデータが部分プログラム間の共有データであるか否かを判定する。このとき、共有データと判定したデータにはその旨の識別子を付加する（ステップＳ５０２）。

【００４３】

次に共有データ識別情報付加部４０２が、コンパイラの内部表現を走査して、ステップＳ５０２で識別子を付加されたデータを検索する。そして検索された共有データに対し、その識別情報として、データ名の先頭に所定の接頭子を付加する（ステップＳ５０３）。なお、この接頭子としては具体的には文字列「___ｓ

ｈｒ＿」などが考えられる。

【００４４】

次に命令列生成部４０３が、コンパイラの内部表現にもとづいて、プログラムの動作を実現する命令列を生成し、当該命令列をコンパイラの内部情報に付加する（ステップＳ５０４）。

【００４５】

次にアセンブリ記述出力部４０４が、コンパイラの内部表現および付加されている命令列にもとづいて、上記プログラムのアセンブリ記述を出力する（ステップＳ５０５）。以上で、コンパイラによるソース記述からアセンブリ記述までの変換処理が終了する。

【００４６】

次に上記装置ではアセンブラが起動され、当該アセンブラにより実現される第２解析部４０５が、ステップＳ５０５でコンパイラのアセンブリ記述出力部４０４から出力されたアセンブリ記述を読み込んで、字句解析をおこなうとともにアセンブラの内部表現へと変換する（ステップＳ５０６）。

【００４７】

次にバイナリ・コード生成部４０６が、アセンブラの内部表現にもとづいてバイナリ・コード（命令コードを含む）を生成し、当該コードをアセンブラの内部情報に付加する（ステップＳ５０７）。

【００４８】

次にオブジェクト出力部４０７が、アセンブラの内部表現および付加されているバイナリ・コードにもとづいて、上記プログラムのオブジェクトを出力する（ステップＳ５０８）。以上で、アセンブラによるアセンブリ記述からオブジェクト・コードまでの変換処理が終了する。

【００４９】

次に上記装置ではリンカが起動され、当該リンカにより実現されるオブジェクト読み込み部４０８が、ステップＳ５０８でアセンブラのオブジェクト出力部４０７から出力されたオブジェクトをリンカの内部表現として読み込む（ステップＳ５０９）。

【００５０】

次に共有データ領域形成部４０９が、リンカの内部表現において、共有データである旨の識別情報（上述の「___ｓｈｒ__」など）を有するデータを検索する。そして、これらの共有データのみから構成される領域（図１に示したｓｈａｒｅｄ　ｄａｔａ　ａｒｅａ１０２）を形成し、リンカの内部表現として付加する（ステップＳ５１０）。

【００５１】

次にメモリ空間構築部４１０が、リンカの内部表現において、残った非共有データのみから構成される領域（同ｄａｔａ　ａｒｅａ１０１）と命令列のみからなる領域（同ｔｅｘｔ　ａｒｅａ１００）とを形成し、リンカの内部表現として付加する（ステップＳ５１１）。

【００５２】

次にアドレス解決部４１１が、リンカの内部表現において、ｔｅｘｔ　ａｒｅａ１００、ｄａｔａ　ａｒｅａ１０１、ｓｈａｒｅｄ　ｄａｔａ　ａｒｅａ１０２の各メモリ領域のアドレス解決をおこなう（ステップＳ５１２）。

【００５３】

次にロードモジュール出力部４１２が、リンカの内部表現にもとづいて、上記プログラムのロードモジュールを出力する（ステップＳ５１３）。以上で、リンカによるオブジェクト・コードの結合が終了し、ソースからロードモジュールまでのプログラムの変換処理が終了する。

【００５４】

以上説明した実施の形態１によれば、複数の部分プログラムにより共有されるデータにはそれと分かる識別情報（具体的には「___ｓｈｒ__」などの接頭子）が付され、リンク時にはまずこの共有データのみが抽出されてｓｈａｒｅｄ　ｄａｔａ　ａｒｅａ１０２が形成された後、残りの非共有データでｄａｔａ　ａｒｅａ１０１が、また残りの命令列でｔｅｘｔ　ａｒｅａ１００が、それぞれ形成される。

【００５５】

このように、ロードモジュールの生成時に共有データと非共有データとがあら

かじめ異なる領域にまとめられているので、実行時には共有データすなわちｓｈａｒｅｄ　ｄａｔａ　ａｒｅａ１０２をキャッシュ非対象領域に、非共有データすなわちｄａｔａ　ａｒｅａ１０１をキャッシュ対象領域に、それぞれ配置するようにすれば、共有データが方々のプロセッサのキャッシュ・メモリに分散する事態が避けられ、これによりキャッシュの一貫性が維持される。

【００５６】

（実施の形態２）

さて、上述した実施の形態１では、リンカによるメモリ空間の構築に先立ってあらかじめ印を付けておいた共有データをいわば取り分けておくようにしたが、これは従来技術のリンカでは、共有データと非共有データとを別々にまとめるという発想がないためである。すなわち、データは単純にソースに記述された順序で結合されてゆくので、ステップＳ５１０で先に共有データだけを取り分けておかないと、続くステップＳ５１１では共有データと非共有データとの混在する領域が形成されてしまうためである。

【００５７】

ただし、従来技術によるリンカも、同一セクション内のデータをそれぞれまとめて一ブロックとし、各ブロックを異なるメモリ領域に配置する機能を備えている。そこで以下に説明する実施の形態２のように、アセンブラのセクション指定疑似命令を利用して、たとえば共有データ部分をセクションＡ、非共有データ部分をセクションＢ、のように指定しておけば、従来技術のリンカによるメモリ空間構築処理の枠内で、共有データと非共有データとをそれぞれ別個のブロックにまとめることができる。

【００５８】

実施の形態２にかかるロードモジュール生成装置のハードウエア構成は、図３に示した実施の形態１のそれと同一であるので説明を省略する。図６は、本発明の実施の形態２にかかるロードモジュール生成装置の構成を機能的に示すブロック図、図７は当該装置におけるロードモジュール生成処理の手順を示すフローチャートである。実施の形態１との差異は、図６には図４に示した共有データ領域形成部４０９に対応する機能部がないこと、これに伴って、図７には図５に示し

たステップＳ５１０に相当する処理がないことである。

【００５９】

また、実施の形態１の共有データ識別情報付加部４０２が、ステップＳ５０３で共有データに所定の接頭子を付加したのに対し、実施の形態２の共有データ識別情報付加部６０２は、ステップＳ７０３で共有データの直前にセクション指定疑似命令、たとえば「．ｓｅｃｔ　”ＳＨＡＲＥＤ　ＤＡＴＡ”」のような一行を挿入する。

【００６０】

そして、実施の形態２によるメモリ空間構築部６０９は、ステップＳ７１０で「ＳＨＡＲＥＤ　ＤＡＴＡ」セクションに所属するデータのみを集めてｓｈａｒｅｄ　ｄａｔａ　ａｒｅａ１０２とし、残りのセクション内のデータはまとめてｄａｔａ　ａｒｅａ１０１、命令列はまとめてｔｅｘｔ　ａｒｅａ１００とする。

【００６１】

この実施の形態２によれば、アセンブラのセクション指定疑似命令により共有データと非共有データとがあらかじめ分離されているので、プログラムの実行時に前者をキャッシュ非対象領域に、後者をキャッシュ対象領域に、それぞれ配置すれば、共有データが方々のプロセッサのキャッシュ・メモリに分散する事態が避けられ、これによりキャッシュの一貫性が維持される。

【００６２】

（実施の形態３）

さて、上述した実施の形態１および２では、プロセッサのメモリ空間上にキャッシュ対象外の領域を設けて、当該領域に共有データを配置することが前提である。したがって、共有データはそもそもキャッシュ・メモリに複写されないのであるが、以下で説明する実施の形態３のように、共有データについてもキャッシュはする設定にしておき、ただデータの読み込み時には手持ちのキャッシュを無効にして、常に主記憶までデータを読みに行くように制御してもよい。

【００６３】

実施の形態３にかかるロードモジュール生成装置のハードウエア構成は、図３

に示した実施の形態１のそれと同一であるので説明を省略する。図８は、本発明の実施の形態３にかかるロードモジュール生成装置の構成を機能的に示すブロック図、図９は当該装置におけるロードモジュール生成処理の手順を示すフローチャートである。実施の形態１との差異は、図８には図４に示した共有データ領域形成部４０９に対応する機能部がないこと、これに伴って、図９には図５に示したステップＳ５１０に相当する処理がないことである。

【００６４】

また、図８には図４に示した共有データ識別情報付加部４０２に代えて、キャッシュ無効化操作付加部８０２が設けられており、これに伴って、図９のステップＳ９０３では接頭子の付加に代えて、このキャッシュ無効化操作付加部８０２によるキャッシュ無効化操作付加処理がおこなわれることになる。

【００６５】

キャッシュ無効化操作の付加とは、具体的には共有データに対するロード命令の直前に、ライトスルー・キャッシュの場合はキャッシュ・メモリ内の当該データのキャッシュ（厳密には、当該キャッシュを含むキャッシュ・ブロック）のインヴァリデート命令、ライトバック・キャッシュの場合はそのフラッシュ命令およびインヴァリデート命令を、それぞれ挿入することである。

【００６６】

この実施の形態３によれば、共有データも非共有データと同様にキャッシュはされるものの、共有データについて各プロセッサのキャッシュ・メモリに蓄積されているそのコピーは事実上使用されない。

【００６７】

すなわち非共有データの読み出しにおいては、キャッシュ・メモリ内にそのコピーがあればプロセッサは当該コピーを読み出すのであるが、共有データの読み出しにあたっては、その直前にインヴァリデート命令を実行して自ら当該コピーを消去してしまうので、キャッシュ・メモリを飛び越して常に主記憶まで当該データの本体を見に行くことになる。これにより、どのプロセッサも共有データについては主記憶上の同一アドレスを読み書きすることになり、キャッシュの一貫性が維持される。

【００６８】

なお、本実施の形態におけるロードモジュール生成方法は、あらかじめ用意されたプログラム（コンパイラ、アセンブラおよびリンカ）がパーソナルコンピュータ、ワークステーションなどの各種のコンピュータ上で実行されることにより実現されるが、このプログラムはＨＤ、ＦＤ、ＣＤ－ＲＯＭ、ＭＯ、ＤＶＤなどのコンピュータで読み取り可能な各種の記録媒体に記録され、当該記録媒体によって配布することができるほか、インターネットなどのネットワークを介して配布することも可能である。

【００６９】

（付記１）複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、

前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定する共有データ判定工程と、

前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに、所定の識別情報を付加する識別情報付加工程と、

前記識別情報付加工程で所定の識別情報を付加されたデータを結合してメモリ空間内のキャッシュ非対象領域に配置するための領域を形成する共有データ領域形成工程と、

を含んだことを特徴とするロードモジュール生成方法。

【００７０】

（付記２）前記識別情報付加工程では、前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに、識別情報として所定の接頭子を付加することを特徴とする付記１に記載のロードモジュール生成方法。

【００７１】

（付記３）前記識別情報付加工程では、前記共有データ判定工程で少なくとも二

つの部分プログラムにより読み書きされるデータであると判定されたデータに、識別情報としてその所属するセクションを指定する情報を付加することを特徴とする付記１に記載のロードモジュール生成方法。

【００７２】

（付記４）複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、

前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定する共有データ判定工程と、

前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに対してキャッシュ無効化操作を付加するキャッシュ無効化操作付加工程と、

を含んだことを特徴とするロードモジュール生成方法。

【００７３】

（付記５）前記キャッシュ無効化操作付加工程では、前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに対する読み出し命令の直前に、当該データのキャッシュのインヴァリデート命令を挿入することを特徴とする付記４に記載のロードモジュール生成方法。

【００７４】

（付記６）複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成プログラムにおいて、

前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定させる共有データ判定工程と、

前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに、所定の識別情報を付加させる識別情報

付加工程と、

前記識別情報付加工程で所定の識別情報を付加されたデータを結合してメモリ空間内のキャッシュ非対象領域に配置するための領域を形成させる共有データ領域形成工程と、

をコンピュータに実行させることを特徴とするロードモジュール生成プログラム。

【００７５】

（付記７）前記識別情報付加工程では、前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに、識別情報として所定の接頭子を付加させることを特徴とする付記６に記載のロードモジュール生成プログラム。

【００７６】

（付記８）前記識別情報付加工程では、前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに、識別情報としてその所属するセクションを指定する情報を付加させることを特徴とする付記６に記載のロードモジュール生成プログラム。

【００７７】

（付記９）複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成プログラムにおいて、

前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定させる共有データ判定工程と、

前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに対してキャッシュ無効化操作を付加させるキャッシュ無効化操作付加工程と、

をコンピュータに実行させることを特徴とするロードモジュール生成プログラム。

【００７８】

（付記１０）前記キャッシュ無効化操作付加工程では、前記共有データ判定工程で少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに対する読み出し命令の直前に、当該データのキャッシュのインヴァリデート命令を挿入させることを特徴とする付記９に記載のロードモジュール生成プログラム。

【００７９】

（付記１１）複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成装置において、

前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定する共有データ判定手段と、

前記共有データ判定手段により少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに、所定の識別情報を付加する識別情報付加手段と、

前記識別情報付加手段により所定の識別情報を付加されたデータを結合してメモリ空間内のキャッシュ非対象領域に配置するための領域を形成する共有データ領域形成手段と、

を備えたことを特徴とするロードモジュール生成装置。

【００８０】

（付記１２）前記識別情報付加手段は、前記共有データ判定手段により少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに、識別情報として所定の接頭子を付加することを特徴とする付記１１に記載のロードモジュール生成装置。

【００８１】

（付記１３）前記識別情報付加手段は、前記共有データ判定手段により少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに、識別情報としてその所属するセクションを指定する情報を付加することを特徴とする付記１１に記載のロードモジュール生成装置。

【００８２】

（付記１４）複数の部分プログラムにより構成され、前記各部分プログラムが複数のプロセッサによりそれぞれ実行されるプログラムのロードモジュールを生成するロードモジュール生成装置において、

前記プログラムに含まれる個々のデータが、前記複数の部分プログラムのうち少なくとも二つの部分プログラムにより読み書きされるデータであるか否かを判定する共有データ判定手段と、

前記共有データ判定手段により少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに対してキャッシュ無効化操作を付加するキャッシュ無効化操作付加手段と、

を備えたことを特徴とするロードモジュール生成装置。

【００８３】

（付記１５）前記キャッシュ無効化操作付加手段は、前記共有データ判定手段により少なくとも二つの部分プログラムにより読み書きされるデータであると判定されたデータに対する読み出し命令の直前に、当該データのキャッシュのインヴァリデート命令を挿入することを特徴とする付記１４に記載のロードモジュール生成装置。

【００８４】

【発明の効果】

以上説明したように本発明によって生成されたロードモジュールの実行時には、複数の部分プログラムにより共有されるデータはそもそもプロセッサのキャッシュ・メモリに複写されないか、あるいは複写はされるが読み出し時には消去されているために、常に主記憶の値が参照・更新されることになり、これによって、同一データを共有する複数の部分プログラムがそれぞれ別個のプロセッサにより実行されても、キャッシュの一貫性が自動的に維持されるロードモジュールを生成することが可能なロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置が得られるという効果を奏する。

【図面の簡単な説明】

【図１】

アンキャッシュ共有データ手法によりキャッシュの一貫性を保障する場合のメモリマップ例を示す説明図である。

【図２】

選択的キャッシュ無効化操作手法によりキャッシュの一貫性を保障する場合のメモリマップ例を示す説明図である。

【図３】

本発明の実施の形態１にかかるロードモジュール生成装置のハードウエア構成の一例を示すブロック図である。

【図４】

本発明の実施の形態１にかかるロードモジュール生成装置の構成を機能的に示すブロック図である。

【図５】

本発明の実施の形態１にかかるロードモジュール生成装置におけるロードモジュール生成処理の手順を示すフローチャートである。

【図６】

本発明の実施の形態２にかかるロードモジュール生成装置の構成を機能的に示すブロック図である。

【図７】

本発明の実施の形態２にかかるロードモジュール生成装置におけるロードモジュール生成処理の手順を示すフローチャートである。

【図８】

本発明の実施の形態３にかかるロードモジュール生成装置の構成を機能的に示すブロック図である。

【図９】

本発明の実施の形態３にかかるロードモジュール生成装置におけるロードモジュール生成処理の手順を示すフローチャートである。

【図１０】

キャッシュ機構を備えた共有メモリ型マルチプロセッサ方式の構成を模式的に示す説明図である。

【図１１】

従来技術のキャッシュ一貫性機構によりキャッシュの一貫性を保障する場合のメモリマップ例を示す説明図である。

【符号の説明】

３００　バスまたはケーブル

３０１　ＣＰＵ

３０２　ＲＯＭ

３０３　ＲＡＭ

３０４　ＨＤＤ

３０５　ＨＤ

３０６　ＦＤＤ

３０７　ＦＤ

３０８　ディスプレイ

３０９　ネットワークＩ／Ｆ

３１０　イーサネット（Ｒ）ケーブル

３１１　キーボード

３１２　マウス

４００，６００，８００　第１解析部

４０１，６０１，８０１　共有データ判定部

４０２，６０２　共有データ識別情報付加部

４０３，６０３，８０３　命令列生成部

４０４，６０４，８０４　アセンブリ記述出力部

４０５，６０５，８０５　第２解析部

４０６，６０６，８０６　バイナリ・コード生成部

４０７，６０７，８０７　オブジェクト出力部

４０８，６０８，８０８　オブジェクト読み込み部

４０９　共有データ領域形成部

４１０，６０９，８０９　メモリ空間構築部

４１１，６１０，８１０　アドレス解決部

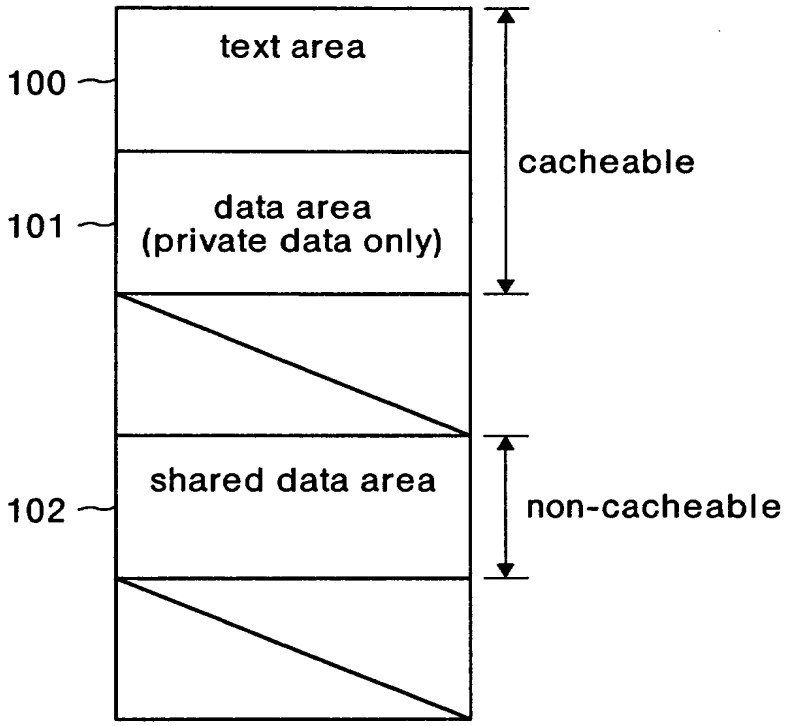
４１２，６１１，８１１　ロードモジュール出力部

８０２　キャッシュ無効化操作付加部

【書類名】　　図面

【図１】

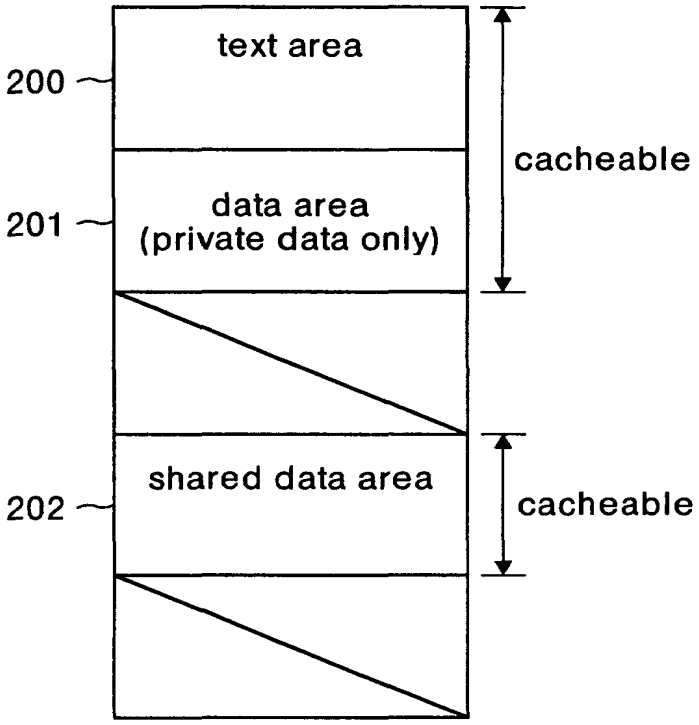
アンキャッシュ共有データ手法によりキャッシュの一貫性を
保障する場合のメモリマップ例



出証特２００２－３１０３５６０

【図２】

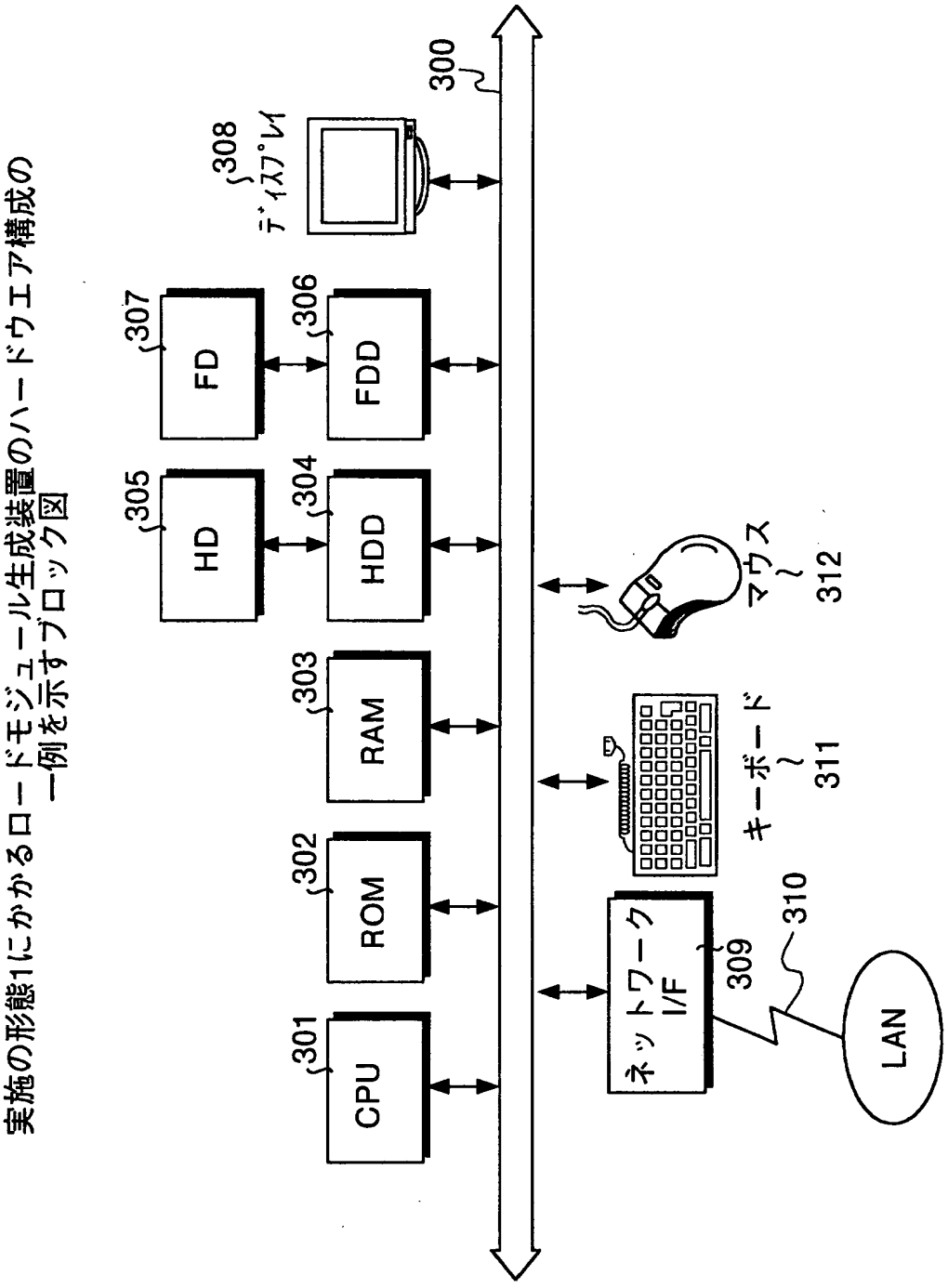
選択的キャッシュ無効化操作手法によりキャッシュの一貫性を保障する場合のメモリマップ例



出証特２００２－３１０３５６０

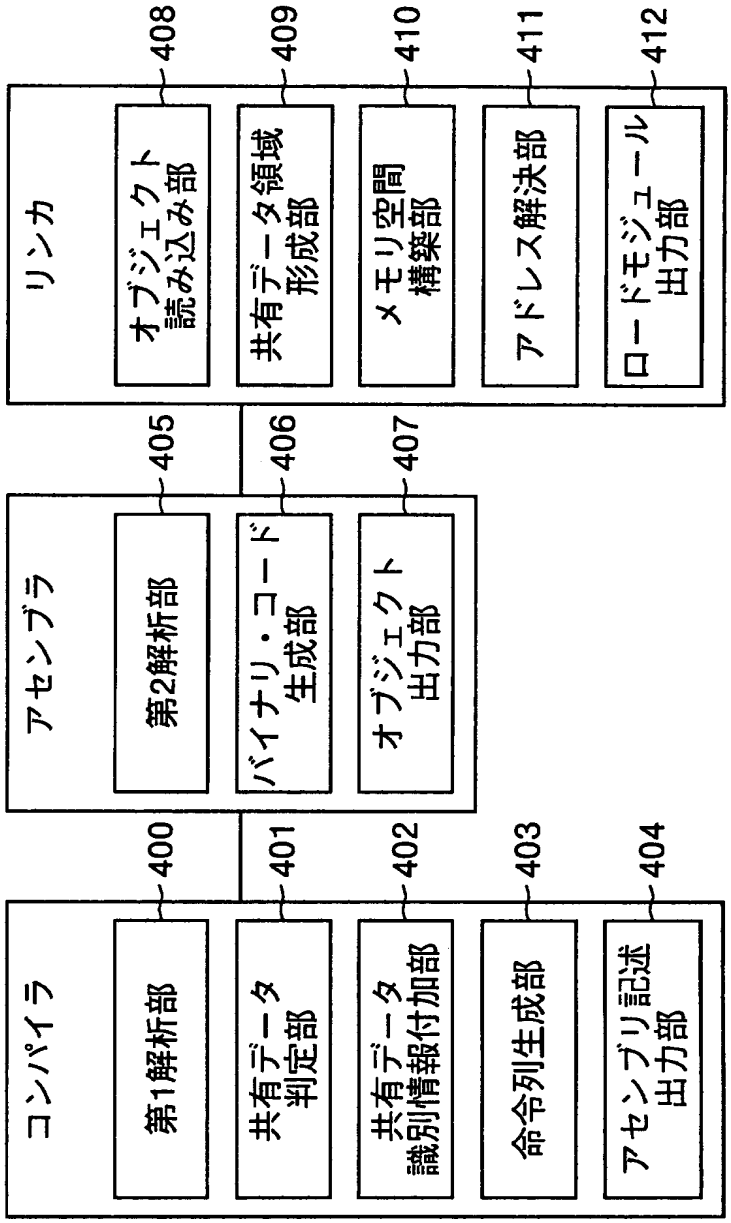
【図３】

実施の形態1にかかるロードモジュール生成装置のハードウエア構成の一例を示すブロック図



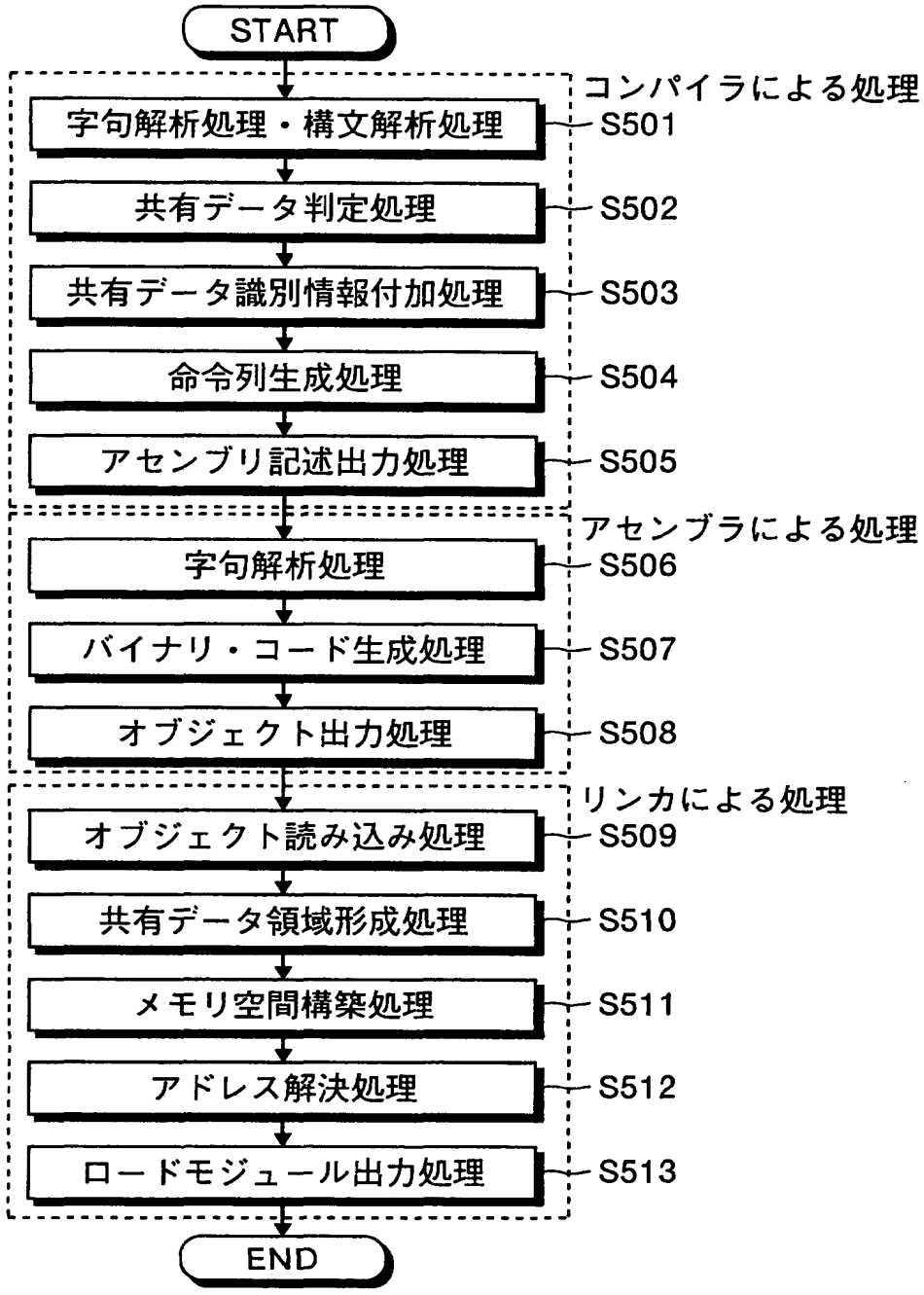
【図４】

実施の形態1にかかるロードモジュール生成装置の構成を機能的に示すブロック図



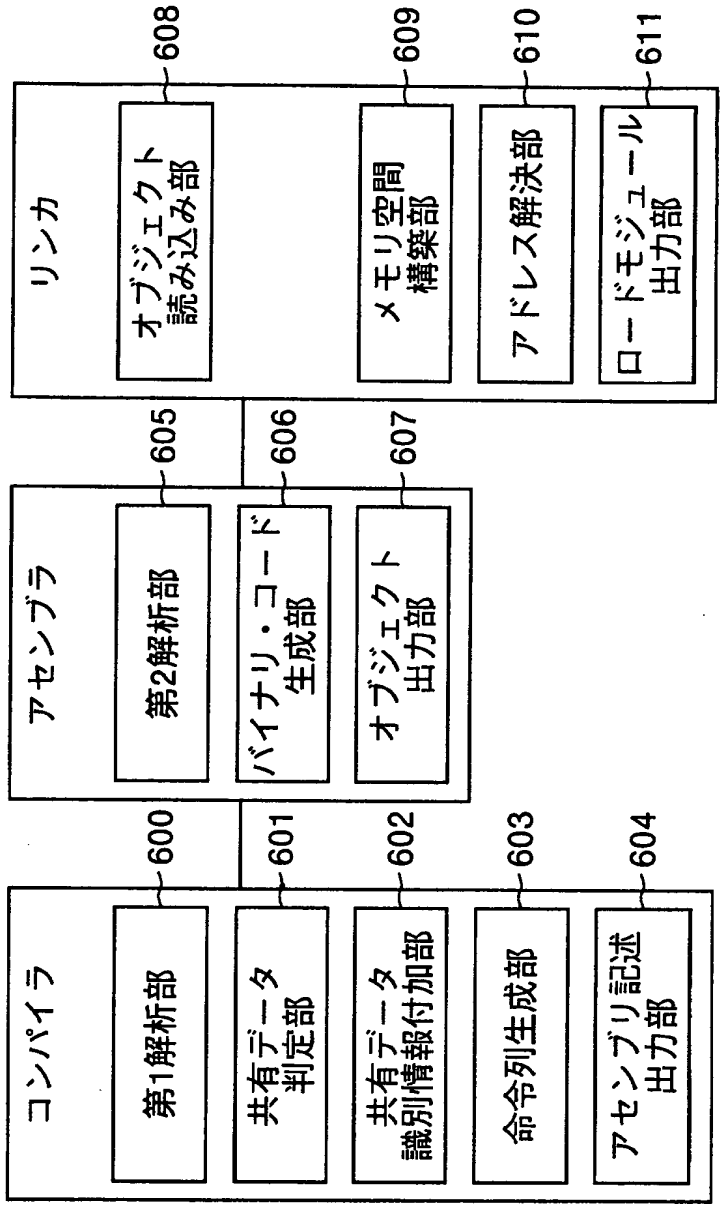
【図５】

実施の形態1によるロードモジュール生成処理の
手順を示すフローチャート



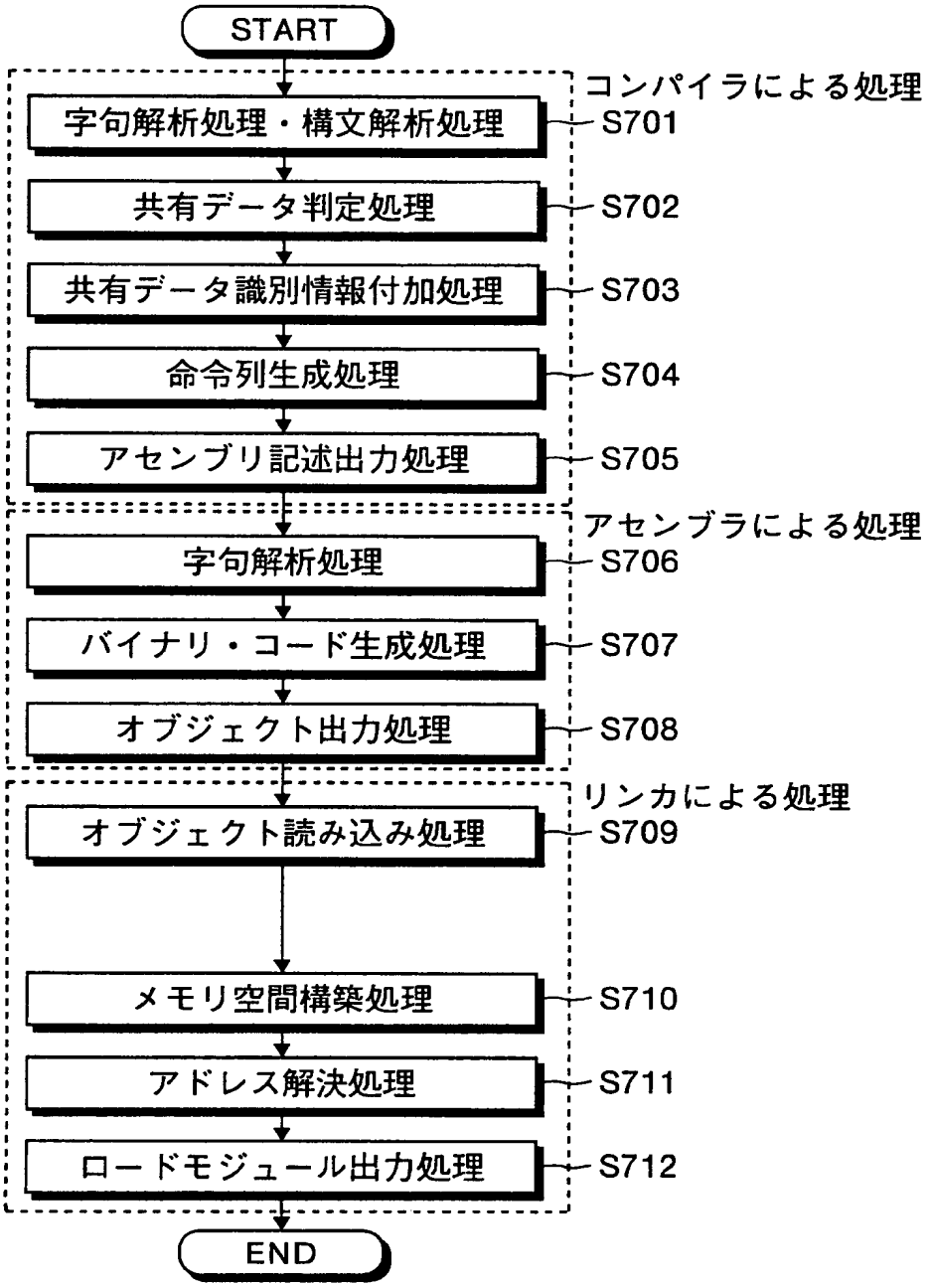
【図６】

実施の形態2にかかるロードモジュール生成装置の構成を機能的に示すブロック図



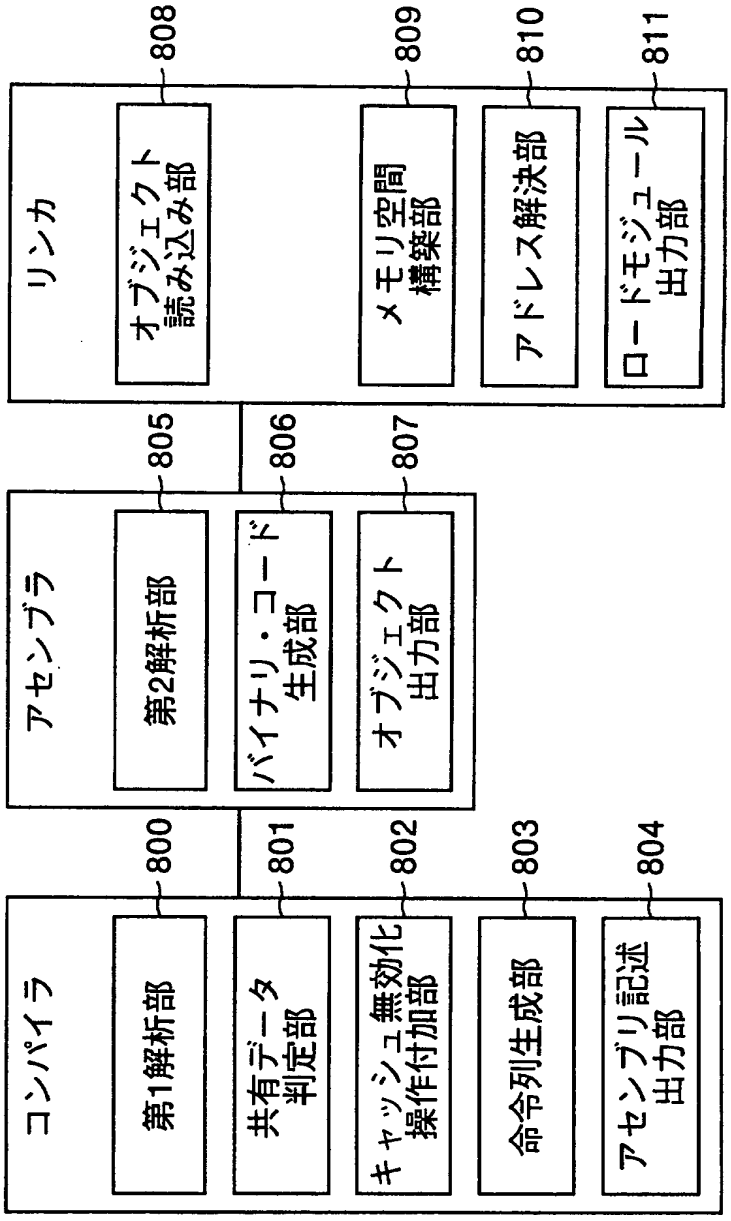
【図７】

実施の形態2によるロードモジュール生成処理の
手順を示すフローチャート



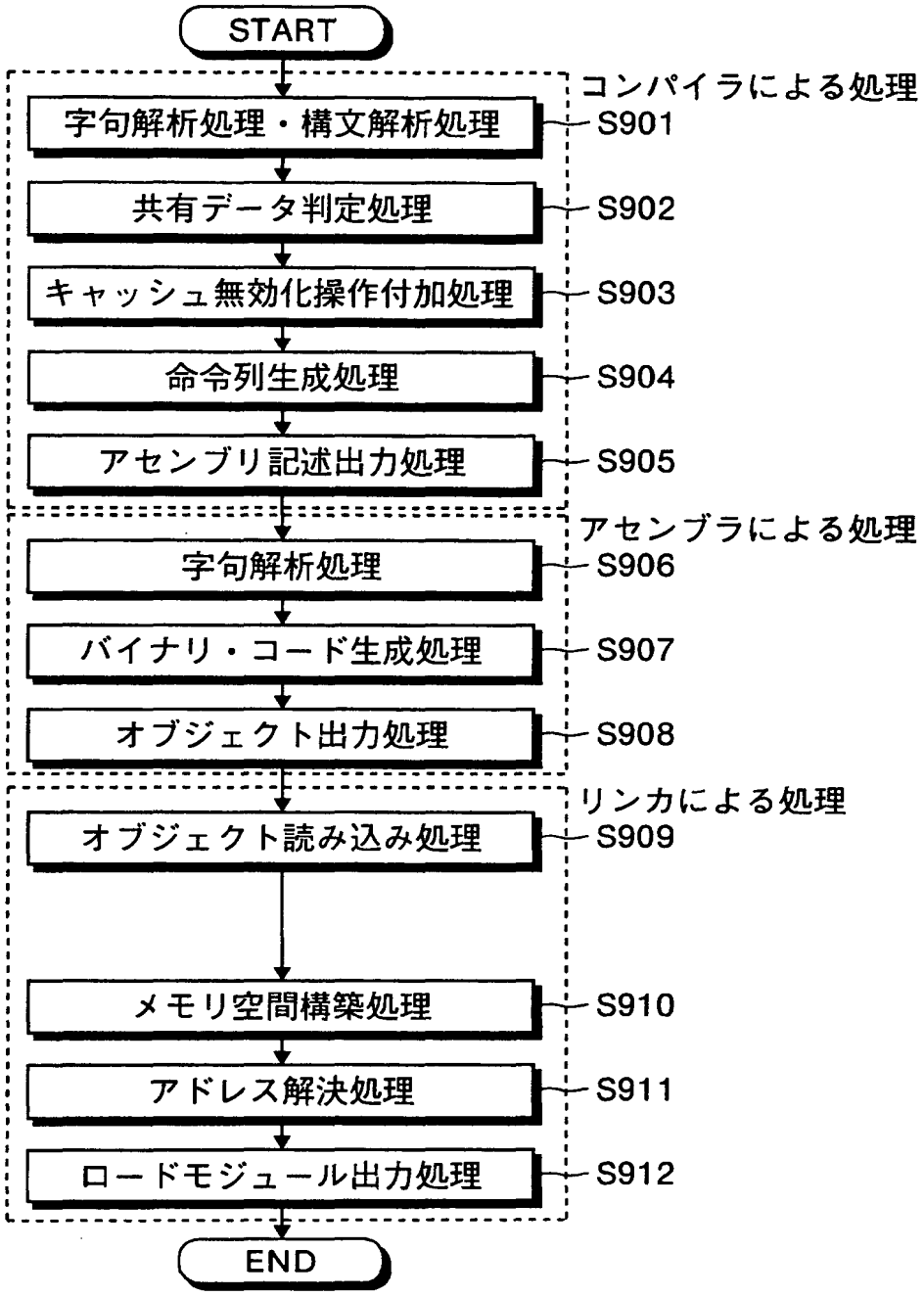
【図８】

実施の形態3にかかるロードモジュール生成装置の構成を機能的に示すブロック図



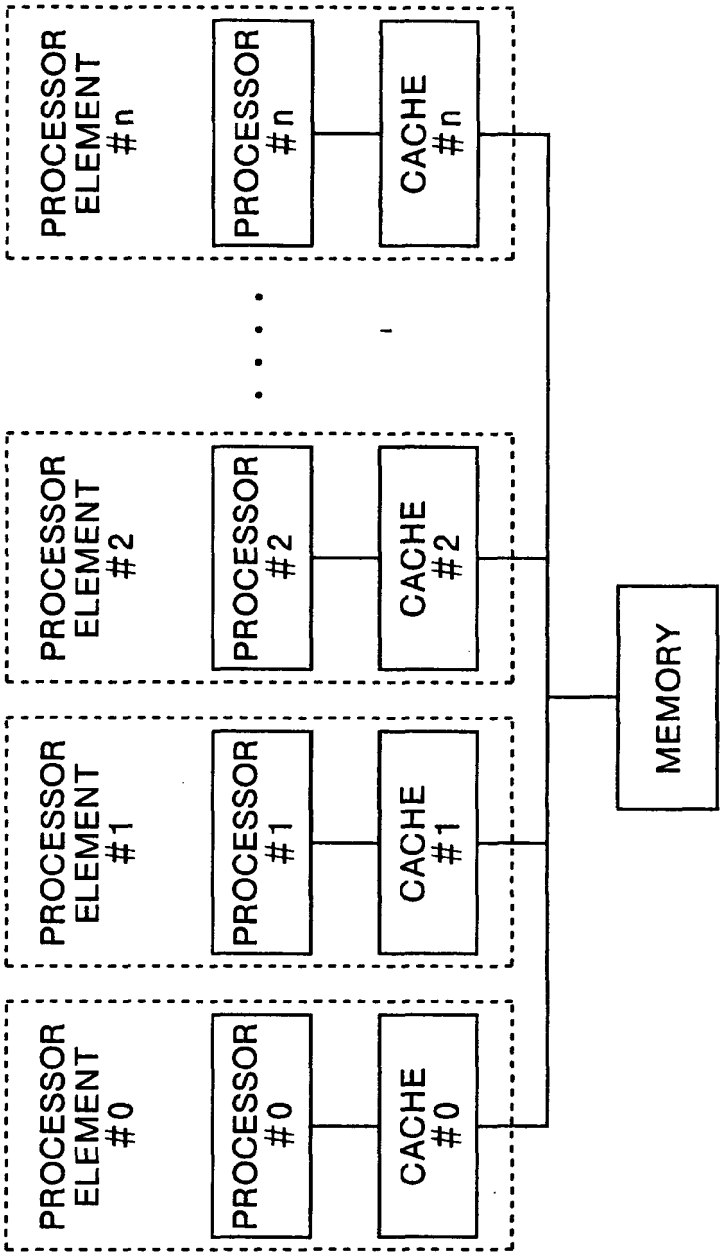
【図９】

実施の形態3によるロードモジュール生成処理の
手順を示すフローチャート



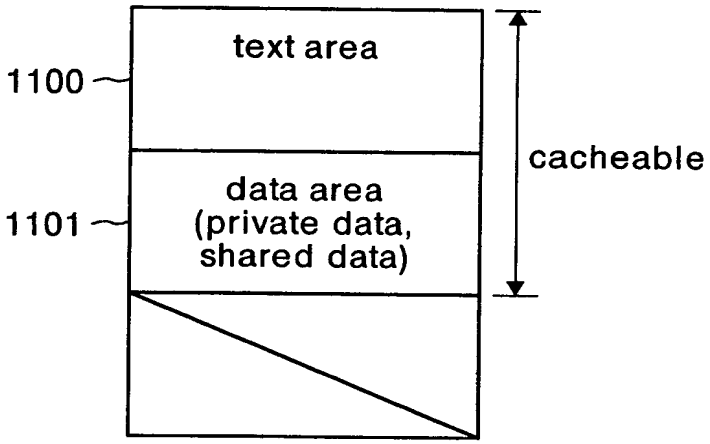
【図１０】

キャッシュ機構を備えた共有メモリ型マルチプロセッサ方式の構成を模式的に示す説明図



【図１１】

キャッシュ一貫性機構によりキャッシュの一貫性を
保障する場合のメモリマップ例



出証特２００２－３１０３５６０

【書類名】　　要約書

【要約】

【課題】　同一データを読み書きする複数の部分プログラムが、主記憶を共有する複数のプロセッサにそれぞれ割り当てられて実行されても、各プロセッサのキャッシュ・メモリと主記憶の間、あるいはキャッシュ・メモリ相互間においてキャッシュの一貫性が自動的に維持されるプログラムのロードモジュールを生成すること。

【解決手段】　コンパイラにより、複数の部分プログラムが共有するデータを特定して所定の識別情報（「＿＿ｓｈｒ＿」などの接頭子）を付加しておき、リンカによるプログラムの結合時に、識別情報の付加されたデータだけを抽出して共有データ領域を形成する。この領域はプログラムの実行時、メモリ空間内でキャッシュ対象外として指定された領域に配置されるので、共有データはキャッシュ・メモリに複写されることがなく、常に主記憶上の唯一の値が参照・更新されることになる。

【選択図】　　図４

出　願　人　履　歴　情　報

識別番号　　〔０００００５２２３〕

１．変更年月日　　１９９６年　３月２６日
［変更理由］　住所変更
住　所　神奈川県川崎市中原区上小田中４丁目１番１号
氏　名　富士通株式会社